Robotic Routers: Algorithms and Implementation

Onur Tekdas, Wei Yang and Volkan Isler *

Abstract

Mobile robots equipped with wireless networking capabilities can act as robotic routers and provide network connectivity to mobile users. Robotic routers provide cost efficient solutions for deployment of a wireless network in a large environment with a limited number of users.

In this paper, we present motion planning algorithms for robotic routers to maintain the connectivity of a single user to a base station. We consider two motion models for the user. In the first model, the user's motion is known in advance. In the second model, the user moves in an adversarial fashion and tries to break the connectivity. We present optimal motion planning strategies for both models. We also present details of a proof-of-concept implementation.

1 Introduction

Suppose a user, working in a large farm or a warehouse, needs continuous network connectivity, perhaps for inventory tracking or surveillance. The user can be a human carrying a mobile device or an autonomous robot. One approach to provide network connectivity to such a user is to deploy static wireless routers and provide connectivity over the entire environment. Even though this solution is viable in some cases, in general maintaining the network (e.g. deployment, replacing batteries) may be costly and inconvenient. Further, since at any given time the set of nodes needed to provide connectivity forms only a small subset of all nodes deployed in the environment, this solution may be too costly in large environments. We believe that robotic routers can provide a better solution for this application: a small number of robots can autonomously deploy and reconfigure themselves to maintain the connectivity of the user. Maintenance operations such as recharging can be performed in an autonomous fashion.

We ran into a similar problem in the Robotics Laboratory at Rensselaer Polytechnic Institute (RPI). The lab is located at the end of a long, rectangular floor (Figure 1). Its wireless network covers only a small portion of this floor which is shared by multiple departments. Therefore, when one of our robots navigates in the halls and needs wireless connection, using other robots to maintain connectivity becomes an appealing solution.

In order to demonstrate the potential utility of introducing mobility, consider the two examples shown in Figure 2. In both examples, circle, square and diamond shapes indicate the base station, user and robotic router, respectively. Suppose the connectivity model is visibility based (i.e. two nodes are connected make it easier to see the connectivity between each other). Note that this model is only for demonstration purposes. In subsequent sections, we consider a general connectivity model which is formalized in Section 3.2. In the left figure, roughly $|V|/3$ stationary routers are needed where $|V|$ is the number of vertices of the environment. However, a single robotic router, which is as

*Corresponding author is Onur Tekdas. Tekdas and Isler are with the Computer Science and Engineering Department at the University of Minnesota. The authors were with the Department of Computer Science, Rensselaer Polytechnic Institute when this work was performed. Emails: {tekdas,isler}@cs.umn.edu,yangw0@gmail.com.
Parts of the results presented in this paper appeared in ICRA 2008 [16].

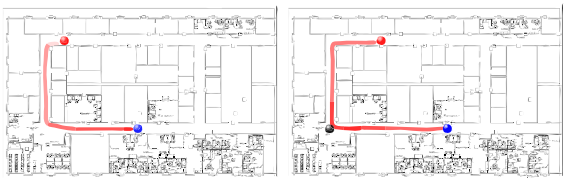

Figure 1: The map and connectivity of the floor where experiments take place. **Left:** The blue circle (in the middle of the bottom corridor) is the stationary blue robot. Fading red circles show the signal strength as the red robot moves to the upper corridor (actual measurements). When the red robot reaches the position shown on the map, the signal strength becomes zero. **Right:** By moving a third robot (black circle) to the position shown on the map, we can reestablish the communication. The colors indicate the signal strength from black to red and black to blue nodes.

fast as the target, suffices for maintaining the target's connectivity. Therefore, in this environment, introducing mobility is beneficial. However, in the right figure, the path of the user is much shorter than the corresponding path of the robotic router. Hence, unless the routers are extremely fast, the number of robotic routers can be as high as the number of static routers.

In this paper, we study the problem of designing motion strategies for a network of robotic routers in order to maintain the connectivity of a single user to a base station. We focus on two user mobility models: In the first model, the user's motion is known in advance. In the second model, the user moves in an adversarial fashion and tries to break the connectivity. We present motion planning strategies for both models. The algorithms are optimal in terms of the duration of the user's connectivity. However, their running times are exponential in the number of robots making them practical for systems with only a small number of robots. Nevertheless, we created a system for maintaining a user's connectivity to a base station for the environment shown in Figure 1. We also present the details of this implementation and an overview of associated design considerations.

The paper is organized as follows. After an overview of related work, we formalize the *Robotic Routers Problem* in Section 3. In Sections 4 and Section 5, we study the robotic routers problem for two different user mobility models. In Section 6 we show the practical feasibility of the algorithms with simulations. In Section 7, we present the details of a real implementation. In Section 8, we present the results of running the motion planning algorithms on the implemented system.

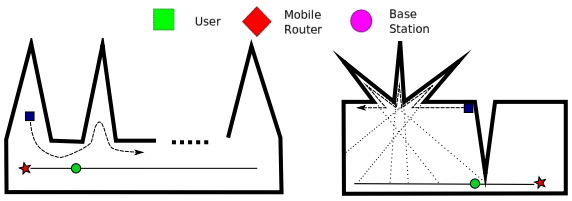

Figure 2: In both figures, we assume a visibility based connectivity model: two nodes can communicate if they can see each other. **Left:** We need one static router per triangular room for stationary case. However, a single robotic router which is as fast as the user can maintain the target's connectivity by following a line. **Right:** As the user visits the spike-shaped rooms, a very fast router can maintain connectivity. However, if the router is not fast enough, the number of necessary mobile routers can be as many as the number of routers in the stationary case.

2 Related work

The problem of maintaining connectivity of a team of robots has been studied for various tasks including rendezvous [8], formation control [11] and flocking [17]. In these problems, all network entries whose connectivity needs to be maintained can be controlled. The difference between these problems and the problem addressed in the present work is that, here we are maintaining the connectivity of a target moving independently from the network.

In this regard, Robotic Routers problem is related to the problem of maintaining the visibility of a single moving target. In [10], authors formulated the problem as a cost minimization problem where they simultaneously penalize the motion of the observer and loss of visibility. In case of known target trajectory, the authors presented a dynamic programming solution which guarantees optimality. For partially-predictable targets, they used a probabilistic motion model for target and presented a solution which maximizes the expected time of visibility. Related work also includes [12] where a sampling-based approach is presented, and [3] where the problem of tracking an evader with a pursuer around a corner is addressed. These problems focus on maintaining the visibility of the target until it disappears for the first time from the field of view of a single robot. In the problems we address, the connectivity model can be more general than visibility and there are additional constraints such as connectivity to the base station. Further, we address the issue of controlling multiple robots.

The problem of maintaining the connectivity of independent users has received significant recent attention. In [14], the authors propose two metrics for characterizing connectivity and present a framework which chooses the best local decision to maintain the connectivity of an independently moving target. The main difference between this work and the work presented in this paper is that here, we are able to provide global guarantees in terms of the duration of the connectivity. We also present a complete implementation. Recent related work also includes [7], where the authors studied the problem of forming a chain of robotic relays and presented an algorithm to control robots along the chain to improve the signal-to-noise ratio.

3 Problem formulation

In this section, we first present the terminology and notation used throughout the paper, and formalize the robotic router problem.

3.1 Definitions and terminology

A *robotic router* is a robot which has wireless communication capability. Robotic routers are subject to communication and motion constraints such as limited communication range and a bounded maximum speed. The *base station* is a static entity to which the *user (or target)* wishes to establish connection. All entities are contained in a shared *workspace* which is represented as a set of points. We refer to the entities that make up the *robotic router network* (the single user, base station and robotic routers) as *nodes*.

In addition to these entities, we use two concepts frequently: *connectivity* and *motion models*. In the mobile router network, two nodes are *connected* if they satisfy the given connectivity requirements which may depend on the position of nodes, communication range of nodes or possible occlusions. The *user is connected* if it is directly connected to the base station or it is connected through point-to-point links in the mobile router network. The *motion model* of the user is discussed in Section 3.3.

3.2 Notation and Assumptions

In this paper, we represent workspace $\mathcal{W}$ as a set of n points. This set contains all possible locations of the nodes (i.e. the user, the base station and the robotic routers). We represent the time domain in unit time steps. All motion models discussed in this paper are represented in the discretized domain as a *trajectory*. Let T be the end time of user motion. The user's trajectory is a sequence u of length T where $u(t)$ is the position of the user at time step t. Similarly, $r_i(t)$ is the position of i^{th} robotic router at time step t. Since the base station does not move, it's trajectory is a constant, b.

A configuration $q = (q_1, ..., q_m)$ is a vector of locations of robotic routers in the mobile router network. As we discretize the time domain, the speed of the user and robotic routers are expressed as step sizes i.e. the distance that a node can move in one time step. In this model, we define the neighbor points that the i^{th} robotic router can move from the point q_j in a single time-step as $N_r(q_j)$ (The subscript r stands for "robot."). To simplify the notation, we assume that all robotic routers have the same speed. One can remove this assumption by defining different neighborhood functions for each robotic router. For the user, we use the neighborhood function N_u. Similarly, $N_c(q)$ is a set of neighbor configurations that can be reached from *configuration* q in one time step. A trajectory r_i is a *valid trajectory* if $\forall t,\ r_i(t+1) \in N_r(r_i(t))$.

Throughout the paper, we do not make any strong assumptions about the connectivity model. We assume that a generic connectivity model is given beforehand. In other words, we are given a matrix A such that $A(i, j)$ is 1 if the mobile router network nodes located at i and j can communicate directly and 0 otherwise. Here, to simplify the notation, we assume that the connection range of all nodes are identical for a particular location. The user located on q_u is *connected* by robots in configuration $q = (q_1, ..., q_m)$ if one of the following holds: (i) $A(q_u, b) = 1$ (ii) $\exists q_i$ s.t $A(q_u, q_i) = 1$ and q_i is connected to b through point-to-point link(s) of type (i) or (ii). Let $q(t)$ be the configuration of mobile router network at time step t. The user is *continuously connected* if it is *connected* in $q(t)$ for all $1 \leq t \leq T$.

3.3 Motion model

In most applications, the workspace, location of the base station, wireless range and speed properties of robotic routers do not change significantly. Hence, the trajectory of the user becomes the most important variable in determining robotic router strategies. In this work we consider two motion models. In the first model we assume that we know the trajectory of the user in advance. This assumption is reasonable for some applications, e.g. in our experiments we control the (user) robots which may have fixed trajectories. In general, a user may be willing to declare its trajectory when requesting the connectivity service.

However, in some cases it is not feasible to know the user trajectory in advance. In such cases, we may consider the worst case trajectory where user tries to disconnect as quickly as possible. This case analysis can give us a guarantee on whether we can connect the user for any possible trajectory or not. We model this scenario as a *pursuer-evader game* where the user tries to break the connection from the mobile router network as quickly as possible. At the same time, the robotic routers try to extend the connection time for as long as possible, preferably infinitely. We call this user motion strategy as *adversarial user trajectory* and the shortest such trajectory as the *shortest escape trajectory*.

3.4 Formulation of the robotic routers problem

In this section, we formalize the robotic routers problem for two motion models: known user trajectory and adversarial user trajectory.

Known user trajectory: Let $\mathcal{W}$ be the workspace, A be the connectivity model, b be the position of the base station, m be the number of robotic routers and u be the trajectory of the user. For each robot r_i, find *a valid robotic router trajectory* such that the user is connected to the base station for the maximum possible amount of time.

Adversarial user trajectory: Let $\mathcal{W}$ be the workspace, A be the connectivity model, b be the position of the base station, m be the number of robotic routers, q_u be the initial location of the user and q be the initial configuration of mobile router network. Find out whether there exists a user escape trajectory for *pursuer-evader game* where the evader wins by breaking connectivity. If it exists, find the *shortest escape trajectory* u. Compute *valid robotic router trajectories*, that maintain the user's connectivity for as long as possible. Since the user can dynamically change its trajectory during execution, the router strategies for the adversarial case must be adaptive. In this paper, we make a design decision and impose the constraint that the robotic router network must remain connected at all times. In other words, each router must remain connected to the base station so that its strategy can be adapted in run time.

In both problems we assume that the number of robotic routers m is given. However, it is easy to obtain the minimum number of routers required for continuous connectivity simply by performing a binary search on m until the continuous connectivity is satisfied.

4 Known user trajectory

In this section, we present *KnownUserTrajectory* algorithm for the robotic routers problem when user trajectory is known a priori. The solution uses dynamic programming to obtain robotic routers' trajectories. Recall that $\mathcal{W} = \{x_1, \ldots, x_n\}$ is the workspace denoted by a set of points. We build a table C where the entry $C(q,t)$ stores the maximum connection time of the user until time t with routers ending in final configuration $q = (q_1, ..., q_m)$. The table size is $n^m \times T$ where T is the length of the user's trajectory. The first m dimensions of the table correspond to router locations. The i^{th} entry in each of these dimensions correspond to location x_i. Using $C(q,t)$, we find robot trajectories which maximize the connection time of the user. We compute the entry $C(q,t)$ iteratively as follows:

$$C(q,t) = \begin{cases} \max\limits_{q' \in N_c(q)} C(q', t-1) + 1 & \text{if } u(t) \text{ is connected by } q \\ \max\limits_{q' \in N_c(q)} C(q', t-1) & \text{otherwise.} \end{cases}$$

$$C(q,1) = \begin{cases} 1 & \text{if } u(1) \text{ is connected by } q \\ 0 & \text{otherwise.} \end{cases}$$

The value: $\max\limits_{\forall q} C(q,T)$ is the maximum connection time for the given user trajectory u. *KnownUserTrajectory* algorithm can be easily modified to return the corresponding robotic router trajectories $r_1, r_2, ..., r_m$ by backtracking as follows. If $\max\limits_{\forall q} C(q,T) = T$, then there exist robotic router trajectories which keep the user continuously connected. Otherwise, we find robotic router trajectories which maximize the connection time of the user. Let $C(q,T)$ be the entry with maximum value among all entries at the last time step (i.e., $q = \arg\max_\rho C(\rho, T)$). In our output trajectory, q will be the final configuration of robotic routers, i.e. $q(T) = q$. Next, we find the entry $C(q', T-1)$ from the previous time-step such that $q \in N_c(q')$ and $C(q', T-1)$ is maximized [1]. In

[1] If there are multiple entries which maximize this value, then there are multiple optimal router trajectories. When this happens, we choose an entry that requires the least amount of movement to reach the configuration in the next time-step.

the output trajectory, q' corresponds to the configuration at time step $T-1$: $q(T-1) = q'$. We continue this computation backwards in time and for each time step, find the robot configuration that maximizes overall connectivity time.

The correctness and optimality of *KnownUserTrajectory* algorithm can be proven directly by induction on t. The running time of the algorithm is determined by the computation time of $C(q,t)$. There are $n^m \times T$ entries. For each entry, computing the maximum value takes $O(n^m)$ time. Hence, the running time is $O(Tn^{2m})$.

5 Adversarial user trajectory

In this section, we present *AdversarialUserTrajectory* algorithm for the robotic routers problem for the case where the user moves in such a way that tries to break the connection as quickly as possible. In contrast to the known trajectory case where the output consists of fixed robot trajectories, the output for the adversarial case is a *strategy* which encodes the appropriate response to every possible user move. The strategy is encoded in Table E where the entry $E[q_u, q]$ corresponds to the positions of the user and robotic routers at an arbitrary time step. Each element of tuple q and q_u is chosen from all possible locations $\mathcal{W}$ of size n. Hence, the dimension of the table is equal to the number of mobile entities in the network $(m+1)$ and the size of the table is n^{m+1}.

The entries of E are filled using the following algorithm:

Algorithm 1 *AdversarialUserTrajectory*

1: $\forall q_u \forall q \;\; E[q_u, q] \leftarrow \infty$
2: $\forall q_u \forall q$
3: ***if*** *robotic router network is not connected* ***then***
4: $\quad E[q_u, q] \leftarrow 0$
5: ***end if***
6: ***for*** $k = 1$ *to* $n^{m+1} - 1$ ***do***
7: $\quad \forall q_u \forall q$
8: $\quad$ ***if*** $\min_{q'_u \in N_u(q_u)} \max_{q' \in N_c(q)} E[q'_u, q'] = k - 1$ ***then***
9: $\quad\quad E[q_u, q] \leftarrow k$
10: $\quad$ ***end if***
11: ***end for***

In the first step (line 1), we initialize all the entries to infinity. At the end of the algorithm, the entry $E[q_u, q]$ gives the length of the shortest escape trajectory starting from user location q_u and robotic routers' initial configuration q. We will show that if an entry $E[q_u, q]$ remains at infinity at the end of the algorithm, then there exists no escape trajectory. In lines 2-5, we set $E[q_u, q] \leftarrow 0$ if the robotic router network (i.e. either user or one of the robots) is not connected. Here, we constrain the entire robotic router network to be connected rather than requiring only the user's connectivity as in the known trajectory case. The reason why we use the specified connectivity condition is due to the dynamic nature of the adversarial user case. Since the user's trajectory is unknown, its location at time t becomes available to the network only at time t. In our system, the

appropriate response is stored at the base station and corresponding motion commands are sent to the robots. Therefore, all robots must remain connected to the base station at all times.

After the initialization steps, we repeat the procedure between lines 7-10 for $n^{m+1}-1$ times. In this procedure, we apply the min-max relation (line 8) to the entire table. Since all configurations where the network is disconnected is already set to 0 in lines 2-5, the maximization is implicitly over configurations where the network remains connected. In each iteration k, we set $E[q_u, q]$ only if the shortest escape trajectory length is k.

With an additional step, we can find if there exists an initial robotic router configuration q corresponding to the initial user location q_u which satisfies the connectivity. If $\exists q,\ E[q_u, q] = \infty$, then we can initialize our robotic routers to configuration q to keep the connectivity uninterrupted. Moreover, if $\forall q_u \exists q,\ E[q_u, q] = \infty$, we can say that m robotic routers are sufficient to maintain the connectivity independent from the initial location and the trajectory of the user.

Next, we explain (i) how we can extract the shortest escape trajectory and the corresponding robotic router trajectories using table E, and (ii) how we can use E to find robotic router trajectories to maintain the connectivity of a user in a system where the trajectory is unknown.

We extract the shortest escape trajectory and the corresponding robotic router trajectories as follows. Let q_u be the initial position of the user: $q_u(1) = q_u$. First, we find the corresponding robotic router positions q which maximize the escape trajectory length, i.e. $q = \arg\max_\rho E[q_u, \rho]$. In the first time step, we will place the robots at locations given by q. Suppose $E[q_u, q] = k$. This means that the length of the escape trajectory is k. We find the next positions of the user and robotic routers by finding the entry $E[q'_u, q'] = k-1$ where $q'_u \in N_u(q_u)$ and $q' \in N_c(q)$. We update the trajectories according to the new positions: $q_u(2) = q'_u$ and $q(2) = q'$. Similarly, we continue for k steps and extract the rest of the shortest escape trajectory and corresponding robotic router trajectories from E. The correctness and optimality of this algorithm is proven in Theorem 2.

In addition to extracting the shortest escape trajectory, we can use E to find robotic router trajectories when we do not know the user trajectory. We assume that the table E is stored at the base station. Further, we require the user to report its current location when it makes a connectivity request to the network. Afterwards, whenever the user makes its next move, it informs the network about its next location. Note that this happens during run-time. For the new location, the base station looks up the new locations of routers which maximize the connection time of the user. Even when the user moves adversarially, the connectivity is ensured for at least the number of steps given by the entry of E that corresponds to the nodes' current locations. As an example, assume that the user starts at position q_u and robotic routers start with configuration q. The user's connectivity is guaranteed for $E[q_u, q]$ steps. If the user's next location is q'_u then we move robotic routers to the configuration $q' = \arg\max_\rho E[q'_u, \rho]$ such that $q' \in N_c(q)$. This guarantees that the user will be connected for at least $E[q'_u, q']$ time-steps.

The running time of the algorithm is $O(n^{3(m+1)})$: there are n^{m+1} entries and for each iteration we scan the entire table which takes $O(n^{m+1})$ time. We iterate this procedure for n^{m+1} times which sums up to the claimed running time.

Theorem 2 *Suppose there exists a shortest escape trajectory such that robotic routers are initially in configuration q and user is at location q_u. Let $e(q_u, q)$ be the length of this trajectory.*

1. *$E[q_u, q] = k$ if and only if the length of the shortest escape trajectory $e(q_u, q)$ is k.*
2. *$E[q_u, q]$ is ∞ if and only if there exist a robotic router trajectory which satisfies continuous connectivity for any possible user trajectory.*

Proof.

Proof of (1): We show that $E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$ by induction on k.

Basis: $E[q_u, q] = 0 \Leftrightarrow e(q_u, q) = 0$ holds due to the initialization step between lines 2-5. If the escape trajectory is of length 0, this means that the user is disconnected in the initial configuration and we set $E[q_u, q] \leftarrow 0$. Similarly, if $E[q_u, q]$ is set to zero in the initialization step, there exists a trivial escape trajectory of length 0.

Inductive step: let us assume that $\forall k,\ E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$ holds. We show that $E[q_u, q] = k+1 \Leftrightarrow e(q_u, q) = k+1$. We prove this statement by showing that both directions of the conditional statement hold.

First we prove: $E[q_u, q] = k+1 \Rightarrow e(q_u, q) = k+1$. For contradiction, suppose that $E[q_u, q] = k+1$ but $e(q_u, q) \neq k+1$. Due to the inductive step we have: $e(q_u, q) \geq k+1$ (Condition 1). This is because, due to the inductive hypothesis, $e(q_u, q) < k+1$ would imply $E[q_u, q] < k+1$, which is a contradiction. When $E[q_u, q]$ is set to $k+1$, due to the min-max relation, following holds: $\exists q'_u \in N_u(q_u),\ \exists q' \in N_c(q)$ such that $E[q'_u, q'] = k$ and $\forall q'' \in N_c(q),\ E[q'_u, q''] \leq k$. From the inductive hypothesis and the inequality: $\forall q'' \in N_c(q),\ E[q'_u, q''] \leq k$, we have $\forall q'' \in N_c(q)$, $e(q'_u, q'') \leq k$. This gives us $e(q_u, q) \leq k+1$ (Condition 2). This is because the user can choose to go to q'_u and follow an escape trajectory of length k afterwards. From conditions (1) and (2), we have $e(q_u, q) = k+1$ which contradicts with the original claim. Thus, $E[q_u, q] = k+1 \Rightarrow e(q_u, q) = k+1$ holds (Condition 3).

Next, we prove: $e(q_u, q) = k+1 \Rightarrow E[q_u, q] = k+1$. Again, for contradiction, let us assume that $e(q_u, q) = k+1$ but $E[q_u, q] \neq k+1$. From the inductive hypothesis, $E[q_u, q] \geq k+1$ holds (Condition 4). Let π be an escape trajectory of length $e(q_u, q) = k+1$ with initial positions of the players given by q_u and q. Let $q'_u \in N_u(q_u)$ be the user location in the second step of π. Since, the escape trajectory length is exactly $k+1$, $\forall q'' \in N_c(q),\ e(q'_u, q'') \leq k$. Because, otherwise robotic router network can increase the connection time by going to q' where $e(q'_u, q') > k$. Moreover, $\exists q' \in N_c(q)$, such that $e(q'_u, q')$ is exactly k (otherwise by going q'_u, user achieves an escape trajectory of length less than $k+1$ which is a contradiction). By the induction hypothesis: $\forall q'' \in N_c(q)$, $E[q'_u, q''] \leq k$, thus applying the min-max relation yields $E[q_u, q] \leq k+1$ (Condition 5). From conditions (4) and (5), we have $E[q_u, q] = k+1$. This is a contradiction with the original claim. Therefore $e(q_u, q) = k+1 \Rightarrow E[q_u, q] = k+1$ holds (Condition 6).

From conditions (5) and (6), the inductive step is proven. Finally, we showed: $\forall k\ E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$.

Proof of (2):

The proof of the second statement is straightforward. $E[q_u, q]$ is either marked as $k \leq n^{m+1}$ or ∞ and the user either has a shortest escape trajectory of length $e(q_u, q) \leq n^{m+1}$ or it can not avoid the connection from the robotic router trajectory. Since the number of iterations in the algorithm can not exceed n^{m+1}, the claim above holds for $E[q_u, q]$. Let us assume that there exists an escape trajectory and its length is: $e(q_u, q) > n^{m+1}$. Since the number of permutations of tuples: (q_u, q) is n^{m+1}, we can find a cycle in the sequence of tuples. However, then we can find a shorter escape trajectory by avoiding the cycle which is a contradiction.

6 Simulations

We demonstrate a practical application of mobile router networks with simulations for the environment is shown in Figure 3. We discretize the hallways into discrete locations almost uniformly (some degeneracy exists near the corners of halls). We construct the connectivity table according to the following rule: if the distance between two locations is less than a fixed distance τ, and they

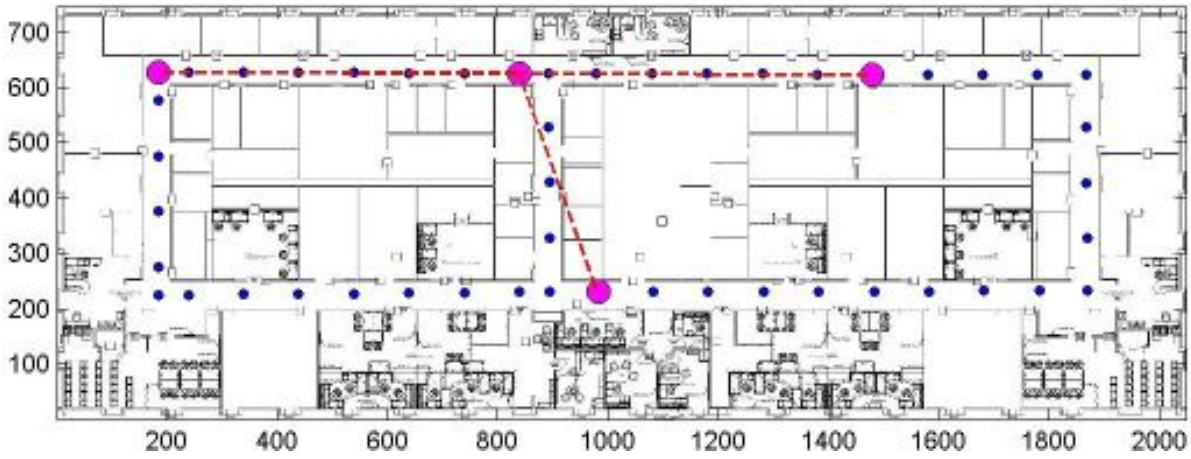

Figure 3: The minimum number of static routers to satisfy the connectivity and coverage constraints is 4. The optimum deployment and its network topology is shown.

are on the same hallway, then these two locations are connected. If two locations are not in the same hallway, their connectivity is based on the geodesic distance between them [2]. To obtain the connectivity threshold τ' for this case, we subtract a fixed penalty from τ for each turn on the path between the two locations. If the geodesic distance between the two locations is less than τ' then these two locations are connected. In the following simulations, the robotic routers are twice as fast as the user. The base station is located at the bottom of the middle vertical hallway.

In order to obtain a baseline, we computed (by enumeration) the minimum number of static routers to cover the environment. It turns out that at least 4 static routers, as shown in Figure 3, are necessary to satisfy coverage and connectivity constraints.

In the following simulations, we start with a network of a single robotic router. For a given (known) user trajectory, we compute the corresponding robot trajectory which keeps the user connected during its trajectory. Next, we find an escape trajectory in which a single robotic router is not sufficient to maintain the connectivity. Finally, we show that two robotic routers are sufficient to keep the user connected whatever initial location or trajectory he chooses. We show how two robotic routers keep the user connected even if the user tries to break the connection. Videos of all simulations are available online [15].

Figure 4 shows the result of our first simulation. The top two figures show the (known) user trajectory and the corresponding robot trajectory computed by our algorithm. We identify the locations of nodes at the critical time steps with time labels. Following figures show snapshots of the connectivity graph of active nodes at these critical time steps. By connectivity graph of active nodes, we indicate the connectivity links (edges) between base station and the user, and the active nodes (vertices) in this connection path.

Figure 5 shows our second simulation. In this simulation, other than the last turn, the user follows the same trajectory as the previous simulation (see left figure). Until this last turn, the robotic router also follows the same trajectory as the previous simulation. However, in the last step, the robotic router can not keep the user connected. The right figure shows the snapshot at the disconnected state.

We find the minimum number of required robotic routers for all possible user trajectories by trying $AdversarialUserTrajectory$ algorithm with increasing number of robotic routers until there exist corresponding robotic router trajectories for all possible initial locations and trajectories of

[2]The connectivity model is inspired by the observation shown in Figure 1 and is also discussed in Section 8.

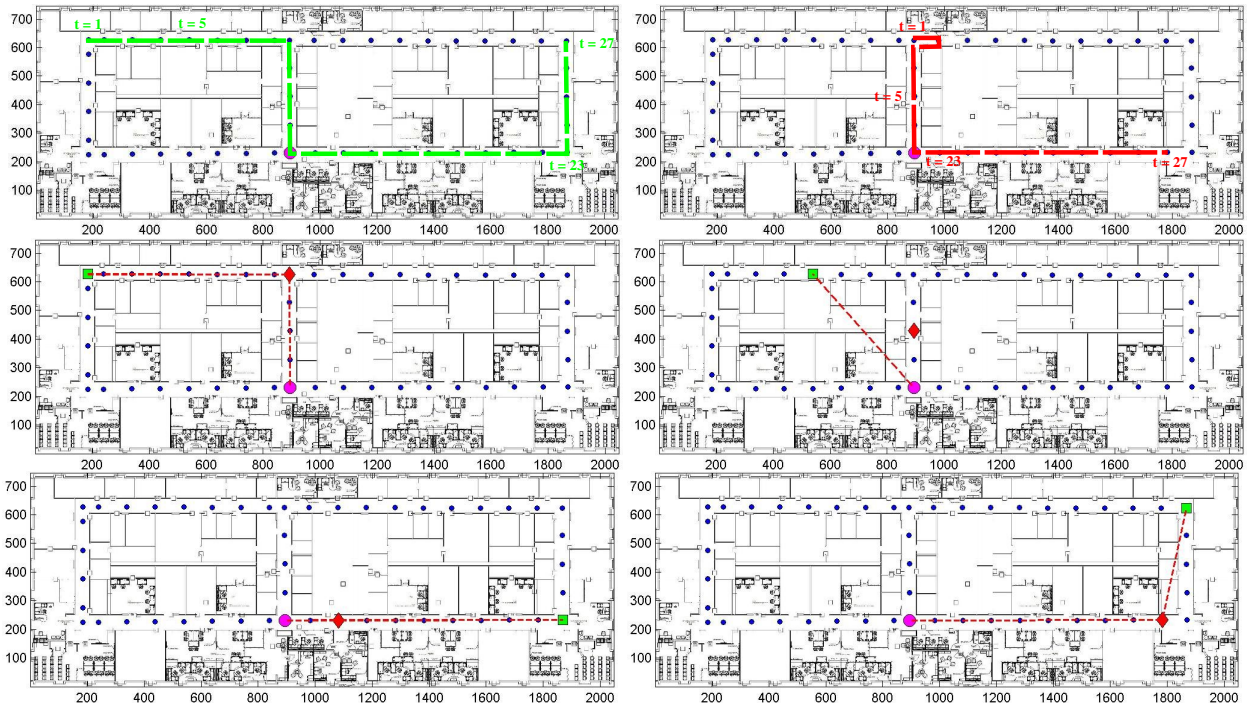

Figure 4: The known user trajectory and corresponding computed robot trajectory are shown in top two figures. The remaining figures show snapshots of the user's connectivity graph (base station - circle with magenta color, robotic router - diamond with red color and user - square with green color). The third figure shows the initial configuration of nodes where the user is connected to the base station through the robotic router. The fourth figure shows the configuration at the time step when the user is directly connected to the base station. The fifth figure shows the configuration when the direct link between the user and base station is broken and connectivity is supplied through the robotic router. The last figure shows the final configuration of nodes.

the user. For this workspace, we found that 2 robotic routers are sufficient to provide a continuous connection.

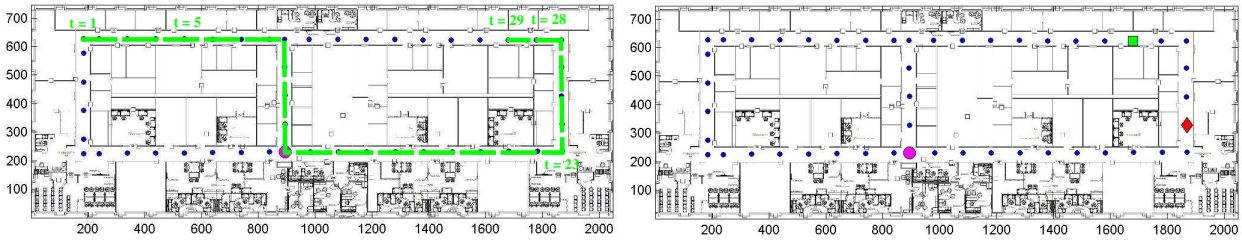

Figure 5: Left figure shows an escape trajectory for the user. Until the last turn, the user and the robotic router follow the same trajectory as in Figure 4. The right figure shows the snapshot from the last step where the user is disconnected.

In the last simulation, relying on the sufficiency of two robotic routers, we solve the known trajectory algorithm with two robotic routers for a path which is not feasible for single robotic router network. The top three figures in Figure 6 show the user trajectory and corresponding robotic router trajectories. The user starts from the top left corner of workspace and completes a cycle by crossing from the middle vertical hall and coming back to the top of the vertical hall. Subsequent figures show connectivity graph of the nodes as snapshots at critical time steps.

7 Implementation

In this section, we present a robotic router system that can maintain the connectivity of a single user on the floor where RPI's Robotics Lab is located. The system utilizes the algorithms presented in this paper. However, for a successful implementation, many technical challenges at the system level must be addressed. These challenges and the design decisions guiding the implementation are presented in this section.

7.1 Overview

The physical components used in this robotic routers system comprise of several identical mobile robots (Figure 7) which are used as robotic routers. In the known trajectory experiment, an additional robot acts as the user. In addition to robots, we use a personal laptop computer as the base station and another personal computer (carried by a human) as the user in the unknown user trajectory implementation. The entities in the network use range limited wireless radios to communicate with one another. The execution of robotic router coordination algorithms (Sections 4 and 5), and most of the data processing is centralized at the base station where the robot tracking and movement commands are also conducted. Each of the robots, as well as the base station, run custom networking applications so that messages can be routed throughout the network.

Although this system can run in any environment, the one used is the rectangular environment with the two rectangle obstacles as shown in Figure 8. The overall system can support both of the real world user motion models: the *known trajectory model* and the *adversarial trajectory model*. In the known trajectory model, the system moves all of the robots in accordance with the pre-computed trajectories based on the known user trajectory. In the adversarial trajectory model, the user is a human walking around with a laptop computer. The user's trajectory is not known to the system (but user reports its next move before each movement). The system is responsible for using

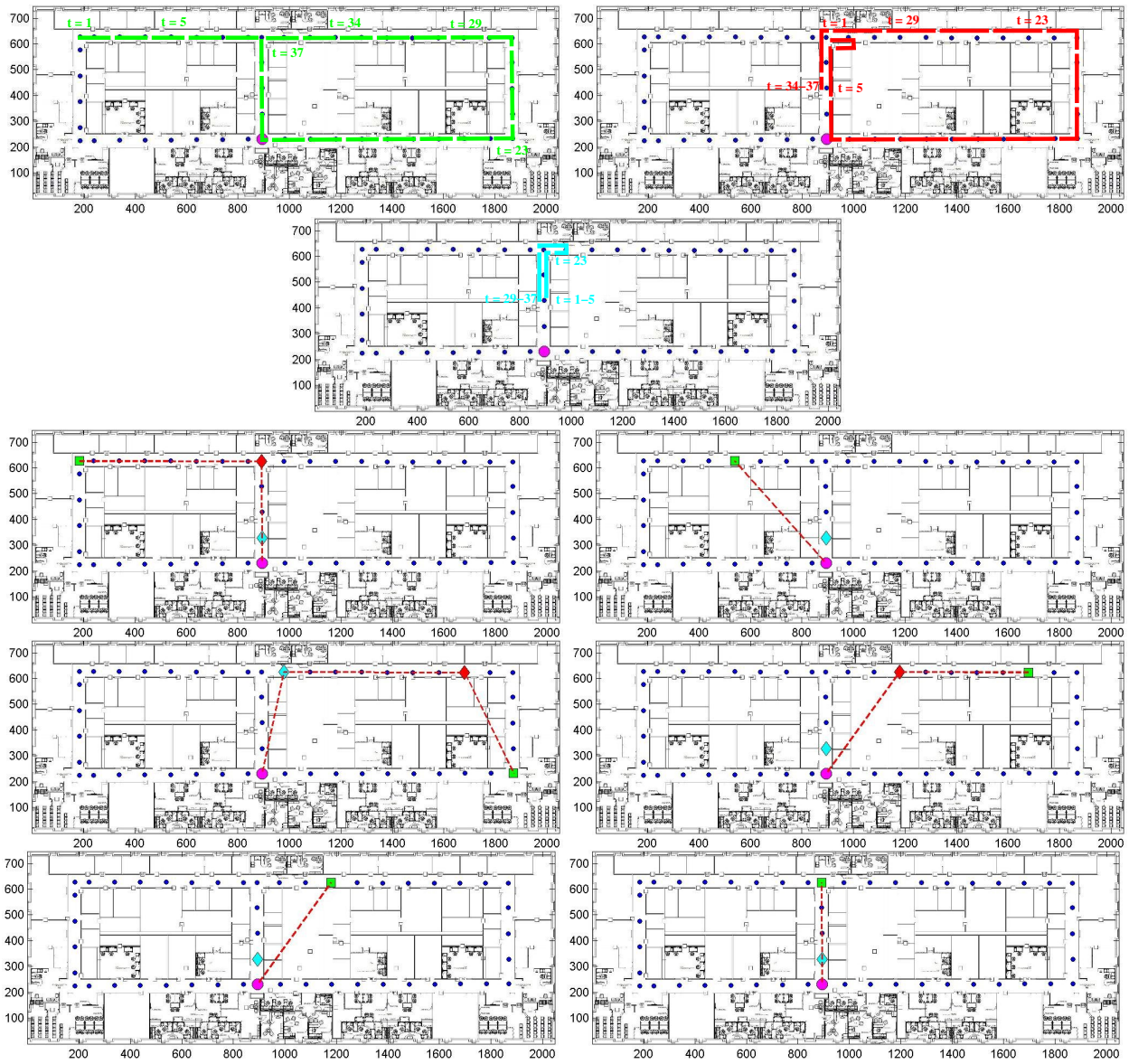

Figure 6: Top three figures show the user trajectory and corresponding robot trajectories. Subsequent figures are snapshots from the solution of the algorithm including the connectivity graph of active nodes (base station - circle with magenta color, two robotic routers - diamond with red and cyan color, and user - square with green color). The first figure on the third row shows the initial configurations of all nodes where the user is connected to the base station through the red robotic router. The second figure on the third row shows the configuration at the time step when the user becomes directly connected to the base station. The first figure on the fourth row shows the configuration when the user is connected to base station through three links. The second figure on this row shows the configuration when the three-link connection reduced to a two-link connection. The first figure on the last row shows the time step when the user is directly connected to the base station. The last figure shows the final positions of the nodes at the last time step.

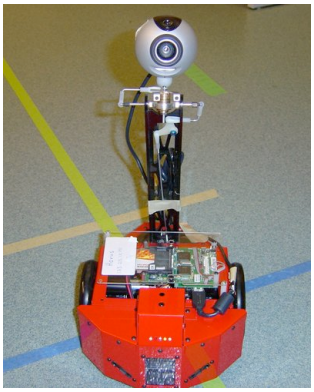

Figure 7: One of the three Acroname Garcia robots used in the implementation

the algorithm from Section 5 to move the mobile robots to the best location in order to maintain the longest connectivity duration.

7.2 Mobile Ad-hoc Network

The IEEE 802.11b wireless standard used only allows nodes to exchange data if they are directly connected. It does not provide any built-in functionality for establishing multi-hop connections between disconnected nodes by routing data through intermediate nodes. Hence, there needs to be a routing protocol to route the data throughout the system.

In the past couple of decades, several mobile ad-hoc network (MANET) protocols have been proposed. Some of them have been implemented and are available to the networking community. OLSR (Optimized Link State Routing) protocol is one of the most well-known MANET protocols [6] [5]. Moreover, it is implemented by the open source community (This implementation is known as OLSRd [1].). Recently, a new protocol, B.A.T.M.A.N, was proposed and implemented to address some of the issues in OLSR (such as topology synchronization and loop routing) [9]. OLSR and B.A.T.M.A.N are known as pro-active protocols where a routing table is constructed for each node and updated regularly even if the node does not have any active transmissions. On the other hand, in reactive protocols, such as Ad-hoc On-demand Distance Vector Routing (AODV) [13], the routing table is constructed on demand. There are some protocols, such as Temporally-Ordered Routing Algorithm routing protocol (TORA), which combine the advantages of pro-active and reactive protocols. In addition, there are protocols (e.g. MosquitoNet [2, 4]) for cases where mobile hosts switch between wireless or wired networks.

In MANET applications, it is assumed that the network entities move independently. Further, these protocols are rather complicated as they are designed to provide networking in a large wireless mesh network. In contrast, in our system, the number of entities is expected to be rather small. More importantly, the mobility of all entities (other than the user) can be controlled. These observations led us to implement a simpler, dedicated protocol. We start with a global communication model. When computing robot trajectories, this model, combined with the estimates of robot locations, is used to find the topology of the network when needed. If the global communication model is conservative (i.e. never gives false positives), we can guarantee that the network topology used for computing robots' paths matches with the actual topology in run-time. On the other hand, depending on the environment, a conservative communication model may significantly underestimate the availability of actual communication links during execution. To address this issue (which is further discussed in Section 8), in the next section we present a broadcasting scheme to send

messages inside the robotic router network.

7.3 Routing Protocol

The algorithms presented in this paper are centralized, in the sense that, a single algorithm decides on the motion of all robots and is run on a single computer (which also acts as the base station). However, due to the dynamic nature of the system (especially in the adversarial case), the base-station must send motion commands to the robots. The base station has direct access to only the robots that are within its communication range. Therefore, to control a robot outside of this range, the base station must send a message through several intermediate robots to the desired robot since there is no global wireless coverage. The problem then becomes a matter of determining which robots to route the message through. Two solutions to routing problem is implemented.

In first implementation, we use the connectivity model which is created for simulations in Section 6. With this connectivity model, the real world system will use a mathematical look up table to determine if two robots are allowed to communicate with each other instead of basing it on the physical environment. Relying on this connectivity model, we extract the connectivity graph of the network. Dijkstra's algorithm is run on this graph to create the shortest routes from the base station to all of the nodes. From here, any information that needs to be transmitted, such as movement commands, can be sent through these paths to the desired nodes. If any of the nodes move, the shortest paths must be recalculated as the connectivity graph could have changed.

The implication of the first routing protocol is that this method may create a discrepancy between the connectivity graph of network that the system thinks it has and the real physical connectivity graph of the network. In a scenario where the mathematical connectivity table is more conservative, this does not present a problem as it can be justified by allowing robots to only communicate when the wireless signal quality is above a very high threshold. The downside is that existing physical connections that could be used are not utilized and this could lead to inefficient networks. However, this does not cause major connectivity problems. On the other hand, if the mathematical model does not take into account some part of the environment and ends up being more relaxed than the physical environment, then major problems can occur since the system expects two robots to be able to talk to each other when in reality, they cannot.

The second solution is to use the broadcasted messages. This is implemented in the intuitive way by enclosing each message in a packet and having each node forward this packet to all of its neighbors until it reaches to the destination node. Each broadcast message is also assigned a unique id and all packets carrying this message would use the same id in their headers. Each node would only process and transmit the first packet it receives for a particular id and would proceed to ignore any further packets of the same id, thereby preventing a situation where the same packet is transmitted throughout the network indefinitely.

After a particular node receives a broadcasted message, such as "send your location to the base station", it must be able to figure out how to reply back if necessary. Depending on the stability of the network, this can be accomplished in two ways. The first way is to simply reply back using the same broadcast mechanism as discussed before. This is a fairly inefficient method since it creates a lot of network traffic with sending even a single message but there are times when this is the only choice, as it will be explained, shortly. Another method is to send the message through the same exact path the broadcasted message took to arrive at the node. The broadcast packets can be modified so that each node appends its unique id to the footer of any packet that it will retransmit to its neighbors. Through this, whenever a node receives a broadcast packet, it can extract the exact path the packet took to reach that particular node.

However, if the network connections are unstable, a situation can occur where a message makes

it through to a node over a weak connection. Since the node will discard any other packets with the same message id, it will ignore a packet which arrived through a longer but stronger path through the network. If the initial weak connection can no longer support data transfer, then the node is stuck in a situation where its only reply path does not work and it is too late to extract paths from any of the duplicate packets.

We implemented and tested both protocols. Since the connectivity model used for simulations matched the actual connectivity model, we decided to use the first protocol in our experiments. This protocol has the advantage of knowing the topology of the network at each time-step. Hence, we can easily calculate network routes to pass messages between nodes. This avoided flooding the network which causes unnecessary traffic.

7.4 Network Localization

One of the assumptions of the system is that the base station knows the location of all of the nodes at all times. This can be justified since the mobile robots are controlled by the base, their locations must be known at all times. For the user, since it is using services provided by the system, it is reasonable to assume that the user must provide its location to the base station in return. However, in real life the location of a robot may not be same as its expected location due to odometry errors. Hence, there must be a way of maintaining each nodes' location at each time step. This system assumes that each node is able to keep track of its own location, for example using vision based or radio signal based techniques. There is no built-in assumption about where and how this information is stored or delivered to the base station.

The implementation of the system allows for two different approaches to maintaining each nodes' location. The first approach simply stores the position and orientation of each node on the base station itself. It leaves the responsibility to robotic routers to go to exact positions and orientation as it is saved in the base station. From a networking perspective, this is the simpler and more efficient of the two approaches as only "move" commands are passed between the base station and the mobile robots. However, this approach brings an overhead to the system where robots need a relocate for correct localization at every time step.

The second approach can be thought of as a distributed system where each node is responsible for keeping track of its own location and orientation. The base station must poll the nodes for their latest positions before each time step. Keeping the location data stored on the robots removes the discrepancy that might occur between the robot position values on the base station and the actual robot locations. However, the advantage of this method also contributes to its disadvantage as well. Updating each node's location at every time step allows for a more accurate system but this also increases the amount of network traffic on the network, especially if broadcasting is used.

The robots used in the experiments do not have sophisticated localization capabilities. In the experiments, we relied on odometry and manually corrected the robots locations when the localization error was too high. This motivated us to use the first approach, since it does not require explicit localization information from the robots.

One simple way of combining the advantages of these two approaches is to use both of them at the same time. By keeping a copy of each nodes' position on the base station and periodically polling each node, a compensation can be easily made between the efficiency of the first method and the accuracy and flexibility of the second one.

8 Experiments

To fully test the algorithm presented in Sections 4 and 5, experiments were ran in the real world using the implemented robotic router system. Both the known and unknown user trajectory cases were tested to see whether the actual implementation would be able to perform as well as the simulated robots.

8.1 Known User Trajectory

In this section, we present an experiment in which the robotic router system maintains the connectivity of the user (a robot), whose trajectory is predetermined, to the base station.

In the experiment, there is a single user robot, two mobile routers and a base station (Figure 8). The user starts off at location 1 with the mobile routers at locations 40 and 69, and the base station at location 49, as seen in Figure 8. The triangles represent the actual location of the robots, with a blue circle at the center. The circles are their target locations that the motion planning algorithm produces at each step. The square represents the base station. Initially, the user is connected to the base station through the mobile router which is at the middle hallway. In all experiments, the dark lines highlight the connectivity graph.

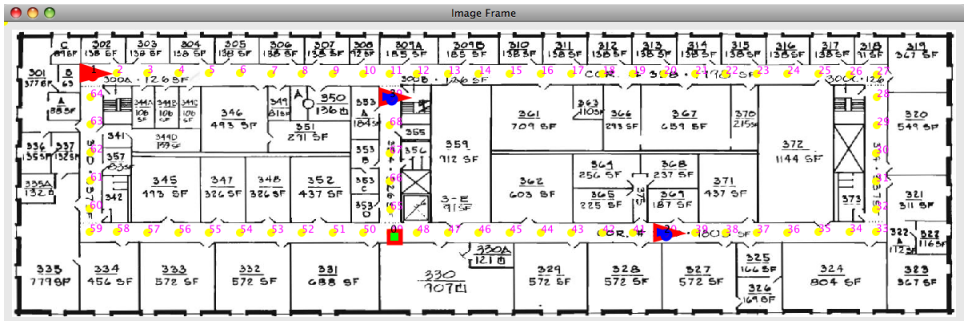

Figure 8: An overhead view of the environment for the real world implementation experiments with the initial starting positions of all of the nodes. The triangles represent the actual location of the robots, with a blue circle at the center. The circles are their target locations that the motion planning algorithm produces at each step. The square represents the base station.

The overall experiment proceeded for 15 minutes which corresponds to 10 steps in which the user (solid red triangle) moved down the left hallway and than head right from the bottom hallway. The mobile router at the right of bottom hallway stayed at the same location during the experiment. This is because, the user is never close enough to utilize its services. However, the other mobile router moved up to the top of middle hallway and moved left to maintain the connectivity of the user (second image in Figure 9). Several steps of the experiment can also be seen in Figure 10 and Figure 11.

Although the experiment was relatively short, it was enough to demonstrate that the overall robotic system is capable of moving three robots simultaneously so that the connectivity is maintained. It also showed that the system can calculate and successfully hand off the user's network connection from a path through a mobile router to being directly connected to the base station. Overall, this experiment was able to demonstrate that the implemented robotic routers system can successfully implement the known user trajectory motion planning algorithm from Section 4. The video of this experiment is available online [15];

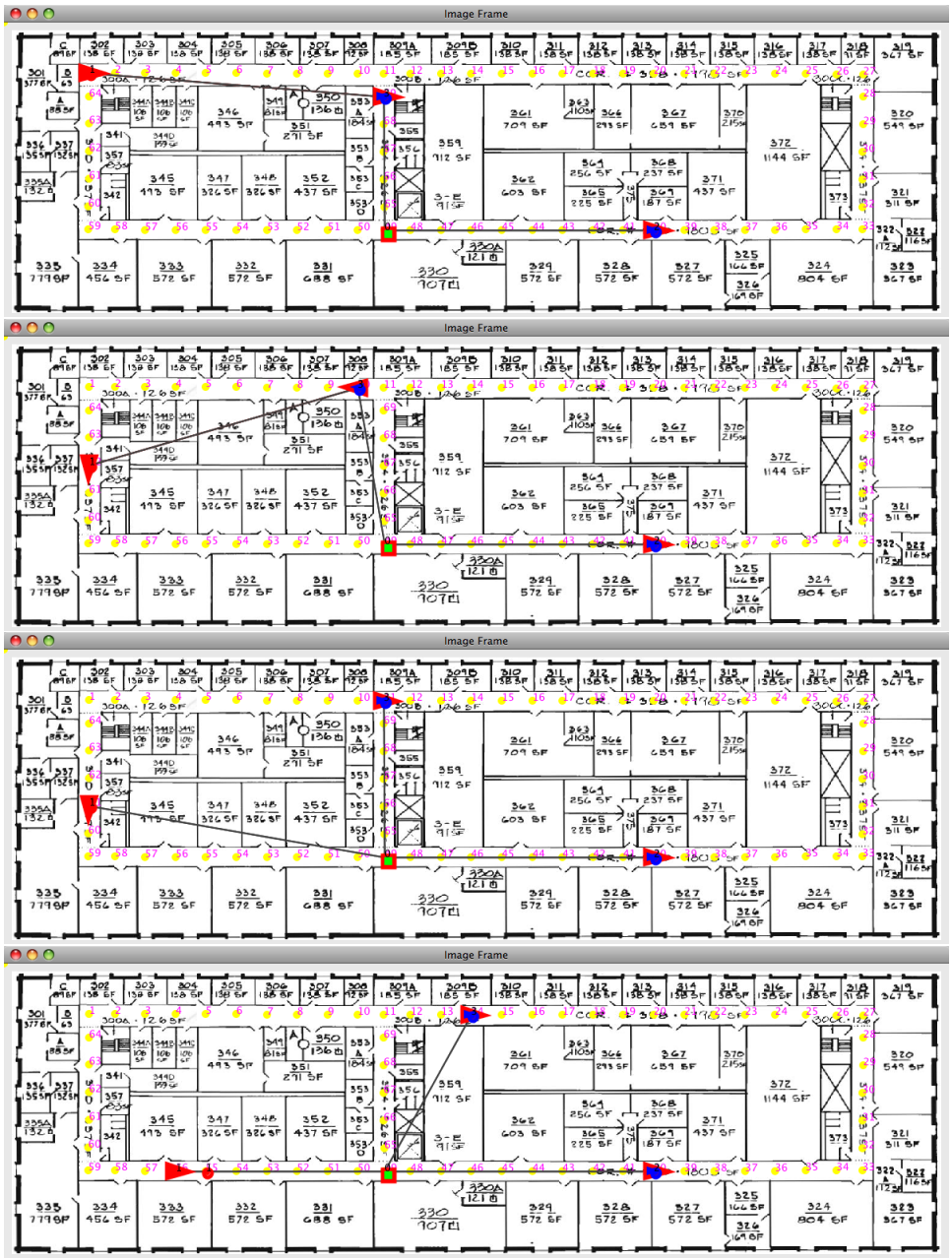

Figure 9: This figure shows various stages of the known user trajectory experiment being conducted using the real world robotic system implementation. The dark lines show the connectivity paths between the nodes. Initially, the user is connected through a mobile router but the path changes as the user moves closer to the base station.

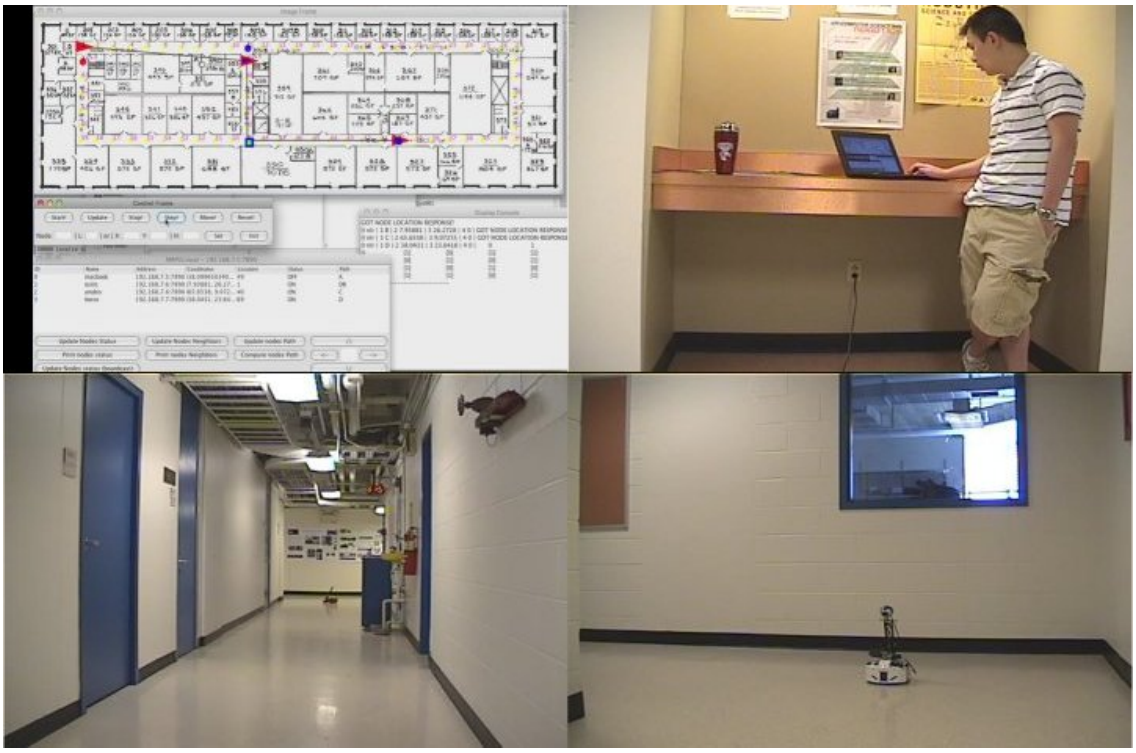

Figure 10: **Known user trajectory experiment-1:** Left top figure shows the implementation graphical user interface (GUI). The GUI shows the initial configuration of the robotic router system. Right top figure shows the Mac base station. Bottom left figure shows the mobile router in the middle vertical hallway. Bottom right figure shows the user which is a robot that controlled remotely by the base station.

8.2 Adversarial User Trajectory

In the adversarial user trajectory experiment, the same environment and experiment setup were used as in the known user trajectory experiment. However, the user is a *human holding a laptop* instead of another mobile router. This new user was allowed to move at the same velocity as the robots but its trajectory was not known ahead of time to the base station or the mobile routers. The only requirement for our design is that the user sends his initial location to the base when he requests a network connection. The user then moves by entering commands locally, such as "r" to signal a desire to move right and similarly for the other three directions. The robotic system must then retrieve the user's desired movement before calculating the next location of the mobile routers. For each next location of the user, we consider the user as an adversarial user and choose the motion strategies for robots which maximize the connection time.

In the experiment, the user first started off at location 16 (in the top hallway) with the mobile routers and base station all at location 49 (Figure 13 and the first image of Figure 12). From here, the user proceeded to move right to the right corner of upper hallway. Initially, the mobile routers did not move because the user was still in range of the base station. However, once the user moved to the limits of its communication range, the first mobile router started to move up to relay messages as seen in Figure 14 and second image of Figure 12. As the user continued to move right, the first mobile router also continued to move up, using itself to maintain the user to base connection. Once this mobile router reached top of middle hallway, it stopped moving because

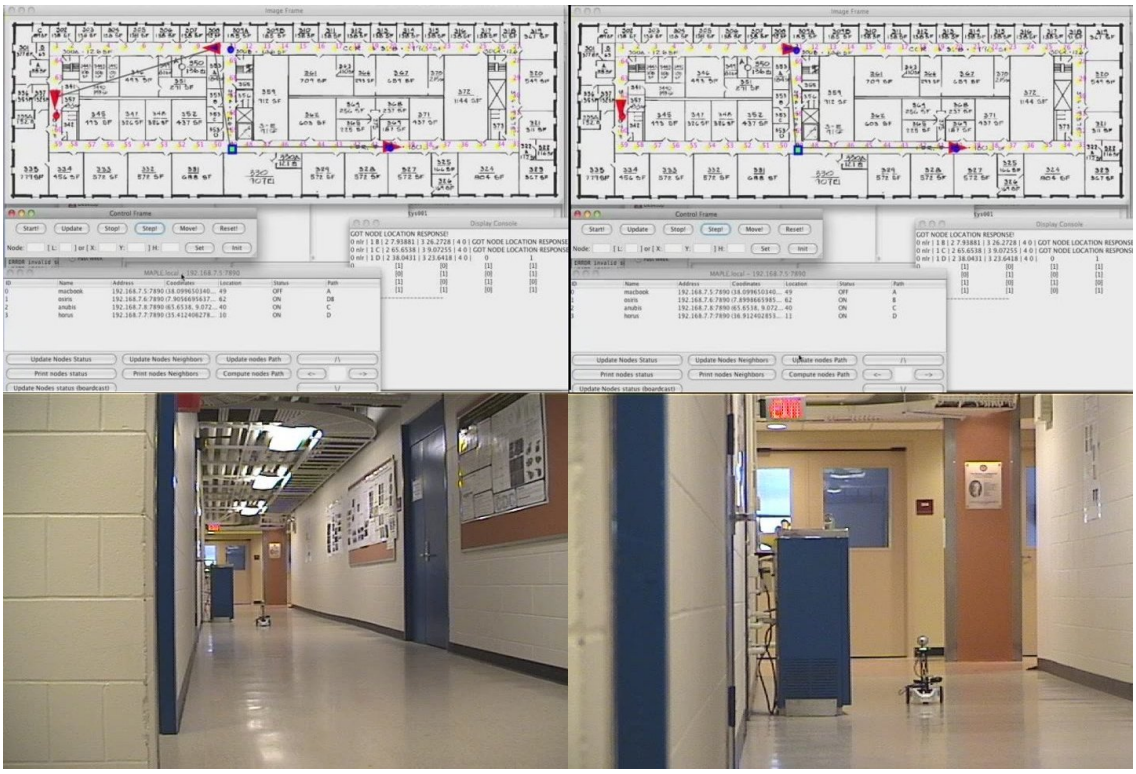

Figure 11: **Known user trajectory experiment-2:** This figure shows the time when the connection path of the user is changed as it moves down the middle of the left hallway. The top row shows the configuration of mobile router network on the GUI while the bottom row shows the corresponding user location at that time step. The left column shows the final time step when user is connected to base station through the mobile router. The right column shows the time step right after the direct connection of user to base station is satisfied.

the user was now visible and connection range was extended. The connection was maintained from mobile routers current position until the user reached to the right end of upper hallway. When user was at the end of hallway mobile router moved one step right to maintain the connectivity (the last image of Figure 12). During this experiment, we acquired perfect connection between the user and base station, and user did not lose any sent data packets.

From this second experiment, the system clearly demonstrated the ability to react dynamically to an unknown user trajectory. It showed that it is able to retrieve the user's next location and using this information, generate the connectivity graph and shortest routes in order to move the mobile routers to their optimal locations. The success of this experiment represents the accomplishment of the initial goal of this paper: using mobile routers to help maintain a multi-hop wireless network connection between a mobile user and base station.

9 Conclusion

In this paper, we addressed the problem of planning the motion of a network of robotic routers. We studied two different user models. In the first model, the user's trajectory is known whereas in the second model the user moves in an adversarial fashion. Even though the algorithms compute optimal solutions, their running times are exponential in the number of robots in the network.

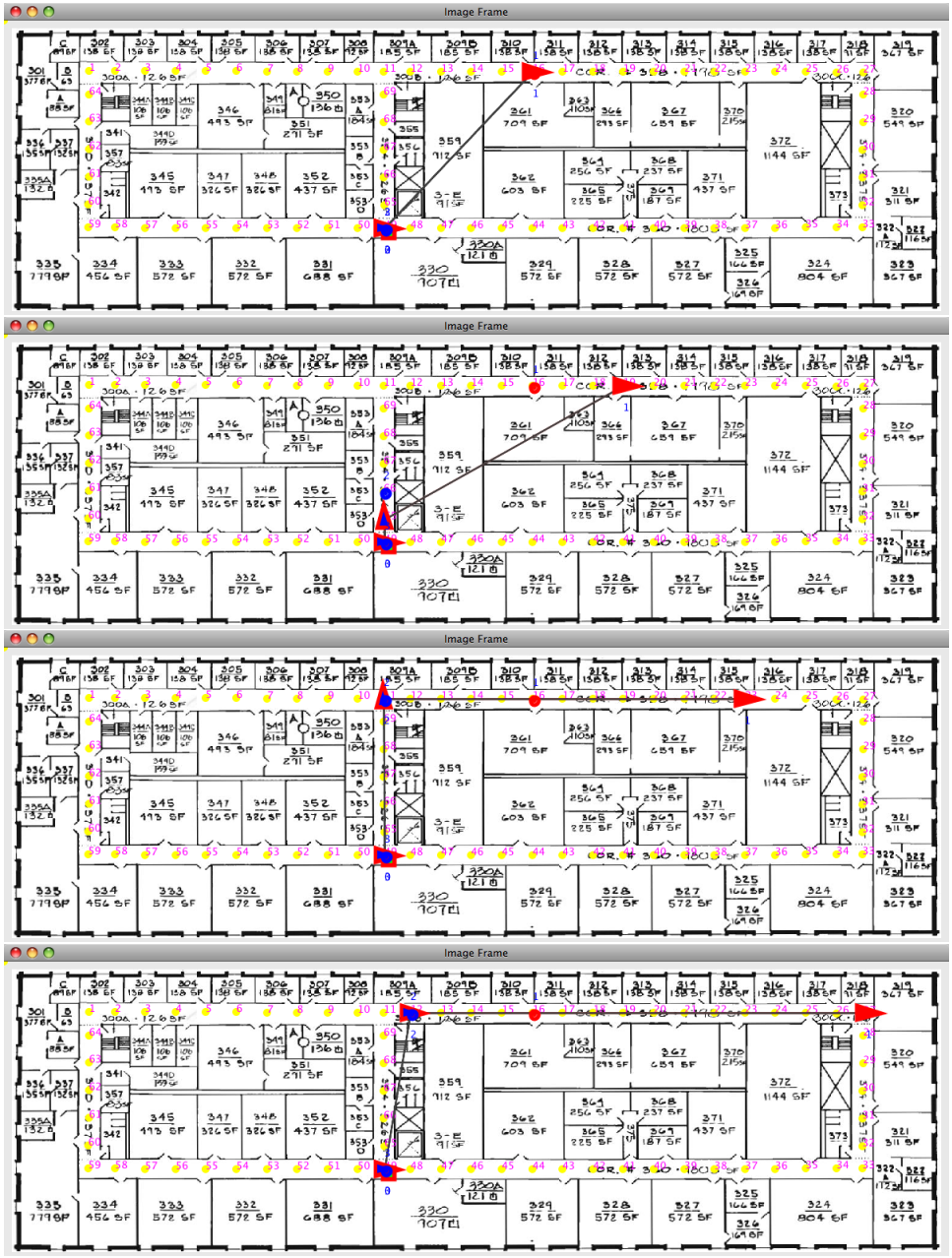

Figure 12: This figure shows various stages of the adversarial user trajectory experiment being conducted using the real world robotic system implementation. The dark lines show the connectivity paths between the nodes. The first figure shows the initial configuration of the mobile router network. The second screen shows a mobile router moving up to maintain connectivity as the user moves to the right. After reaching the top, the mobile router remains stationary until the user reaches the end of the hallway. When this occurs, the router starts to move right in an attempt to maintain connectivity between the user and base station.

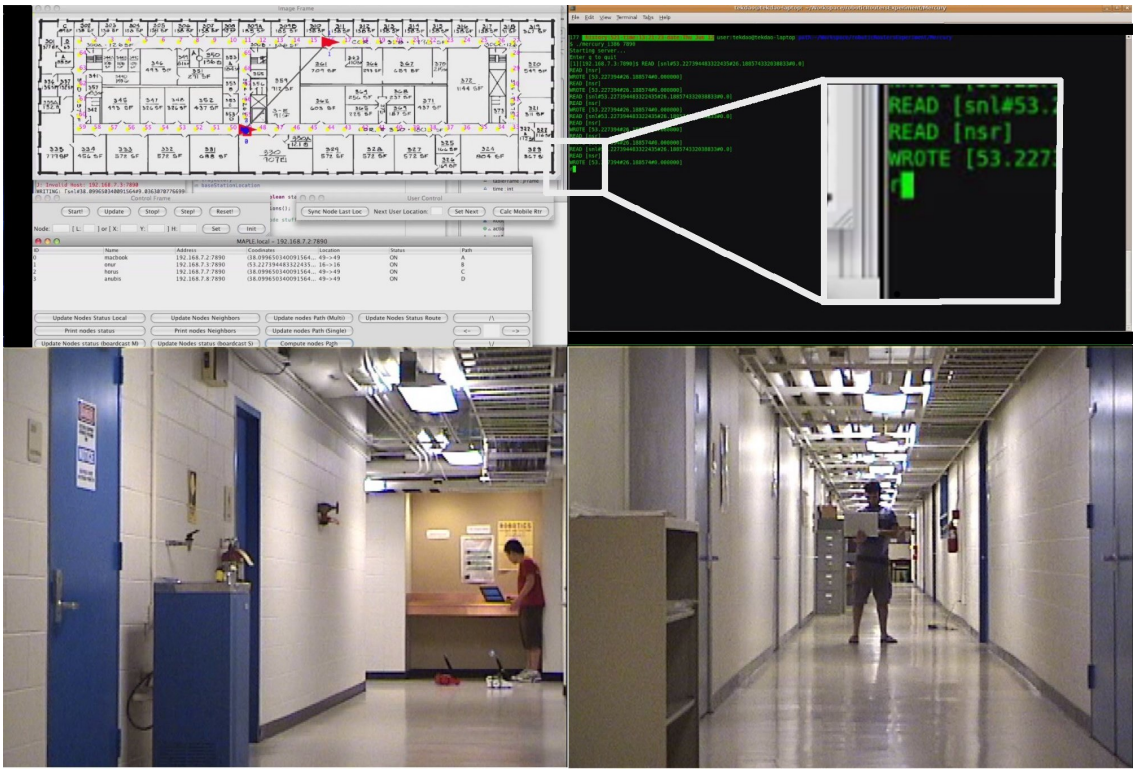

Figure 13: **Unknown user trajectory experiment-1:** This figure shows the initial locations of the mobile router network. In this experiment, the mobile router network keeps the connectivity of an adversarial user (laptop) who requests wireless connectivity and sends acknowledgment of its move in each time step. The top left figure shows the GUI, the top right figure shows the message sent from user, the bottom left figure shows the robots and base station, and the bottom right figure shows the user (see also Extension 1).

On the theoretical front, we are working on improving the running time of the algorithms. We also presented a complete system implementation. Future work for the implementation includes incorporating more robust communication models between the nodes. The algorithms presented in this paper can be extended to the case of multiple users in a fairly straightforward fashion. However, their running times (especially for the adversarial case) become prohibitively costly. Our future work also includes designing efficient algorithms to support multiple users.

10 Acknowledgment

This work is supported in part by NSF IIS-0745537, NSF CCF-0634823 and NSF CNS-0707939. The authors would like to thank Eric Meisner for his help in generating Figure 1.

References

[1] OLSRd. http://www.olsr.org/.

Figure 14: **Unknown user trajectory experiment-2:** This figure shows the second step of the user. The user continues his movement in the right direction (the input "r" shown in top right figure). To keep him connected, the mobile router moves one step forward (see also Extension 1).

[2] Mary G. Baker, Xinhua Zhao, Stuart Cheshire, and Jonathan Stone. Supporting mobility in mosquitonet. In *ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, pages 11–11, Berkeley, CA, USA, 1996. USENIX Association.

[3] S. Bhattacharya, S. Candido, and S. Hutchinson. Motion strategies for surveillance. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[4] Stuart Cheshire and Mary Baker. A wireless network in mosquitonet. *IEEE Micro*, 16(1):44–52, 1996.

[5] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR), October 2003. RFC 3626.

[6] Thomas Heide Clausen, Gitte Hansen, Lars Christensen, and Gerd Behrmann. The optimized link state routing protocol, evaluation through experiments and simulation. In *In Proceeding of Wireless Personal Multimedia Communications. MindPass Center for Distributed Systems, Aalborg University, Fourth International Symposium on Wireless Personal Multimedia Communications*, 2001.

[7] Cory Dixon and Eric W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. In *First International Conference on Robot Communication and Coordination (ROBOCOMM)*, Athens, Greece, 15-17 Oct. 2007.

[8] A. Ganguli, J. Cortes, and F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, November 2006. To appear.

[9] M. Ikeda, G. De Marco, L. Barollif, and M. Takizawa. A BAT in the lab: Experimental results of new link state routing protocol. *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, pages 295–302, March 2008.

[10] S. M. LaValle, H. H. González-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 731–736, 1997.

[11] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *IEEE Transactions on Automatic Control*, 49(4):622–639, 2004.

[12] Rafael Murrieta-Cid, Benjamín Tovar, and Seth Hutchinson. A sampling-based motion planning approach to maintain visibility of unpredictable targets. *Auton. Robots*, 19(3):285–300, 2005.

[13] C. Perkins, E. Belding-Royer, and S. Das and. Ad hoc on-demand distance vector (AODV) routing, July 2003. RFC 3561.

[14] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1525–1530, 2008.

[15] O. Tekdas. Robotic routers network simulation videos. http://rsn.cs.umn.edu/index.php/Robotic_Routers.

[16] O. Tekdas and V. Isler. Robotic routers. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1513–1518, 2008.

[17] M. M. Zavlanos and G. J. Pappas. Flocking while preserving network connectivity. In *IEEE Conference on Decision and Control*, 2007.